**Kubernetes Services - Study Notes (Day 35)**

**Course: Complete DevOps Course**  
**Instructor: Abhishek**

**1. Introduction**

**Why Use Kubernetes Services?**

* **Challenges Without Services:**
  + Pod IPs change when restarted (ephemeral nature).
  + No built-in load balancing for replicated Pods.
  + No direct external access to applications.
* **How Services Help:**  
  ✅ Enable load balancing.  
  ✅ Facilitate service discovery (via labels/selectors).  
  ✅ Provide external access when required.

**2. Key Concepts**

**1. Service Discovery & Load Balancing**

* **Labels & Selectors:**
  + Services track Pods using **labels** (e.g., app: payment).
  + Example:
  + # Pod Label
  + metadata:
  + labels:
  + app: payment
  + # Service Selector
  + spec:
  + selector:
  + app: payment
* **How It Works:**
  + A Service directs traffic to **all Pods** with matching labels.
  + If a Pod fails, ReplicaSet replaces it with a new Pod with the **same label**.

**2. Types of Kubernetes Services**

| **Service Type** | **Use Case** | **Accessibility** |
| --- | --- | --- |
| **ClusterIP** | Internal communication (default) | Within the Kubernetes cluster |
| **NodePort** | Internal + VPC access | Accessible via worker node IPs |
| **LoadBalancer** | External access via cloud providers | Available to public users (e.g., internet) |

**3. Hands-On: Creating a Kubernetes Service**

**Step 1: Define a Deployment (deployment.yaml)**

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

**Step 2: Define a Service (service.yaml)**

**Option A: ClusterIP (Internal Communication)**

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

type: ClusterIP

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

**Option B: NodePort (Internal + VPC Access)**

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

type: NodePort

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

nodePort: 30007 # Optional (Range: 30000-32767)

**Option C: LoadBalancer (Public Access)**

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

type: LoadBalancer

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

**Step 3: Apply Configuration**

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

**Step 4: Verify the Deployment**

kubectl get svc # List services

kubectl describe svc nginx-service # Get service details

**4. Essential Kubernetes Commands**

| **Task** | **Command** |
| --- | --- |
| List all Services | kubectl get svc |
| Get detailed Service info | kubectl describe svc <name> |
| Access ClusterIP Service | curl <cluster-ip>:80 (within cluster) |
| Access NodePort Service | curl <node-ip>:30007 (from VPC) |
| Access LoadBalancer Service | curl <external-ip>:80 (public) |

**5. Interview Questions**

**Q1: Why use Services instead of direct Pod IPs?**

* Pod IPs are **ephemeral** and change when Pods restart.
* Services provide **stable endpoints** and load balancing.

**Q2: How does Kubernetes Service Discovery work?**

* Services utilize **labels and selectors** to track Pods dynamically.

**Q3: What is the difference between NodePort and LoadBalancer?**

* **NodePort:** Exposes the service on worker node IPs for internal/VPC access.
* **LoadBalancer:** Creates a cloud-managed Load Balancer for public access.

**6. Summary**

* **Kubernetes Services solve:**
  + Load balancing across Pods.
  + Service discovery using labels/selectors.
  + External and internal service exposure.
* **Always use Services** instead of directly accessing Pod IPs in production.

🚀 **Assignment:**

1. Deploy an application with a **ClusterIP** Service.
2. Modify the Service type to **NodePort** and test access.
3. (Optional) Deploy a **LoadBalancer** Service on AWS/GCP.

📌 **Pro Tip:** Use kubectl expose for quick Service creation:

kubectl expose deployment nginx-deployment --port=80 --type=NodePort

📢 **Feedback?** Comment below! 👍 **Like & Share** if this helped!

**End of Notes** 🎉

# \*\*Kubernetes Services - Study Notes (Day 35)\*\*

\*\*Course: Complete DevOps Course\*\*

\*\*Instructor: Abhishek\*\*

---

## \*\*1. Introduction\*\*

### \*\*Why Services?\*\*

- \*\*Problem without Services:\*\*

- Pod IPs change when Pods restart (ephemeral nature).

- No load balancing between Pod replicas.

- No external access to applications.

- \*\*Solution:\*\* Kubernetes \*\*Services\*\* provide:

✅ Load balancing

✅ Service discovery (via labels/selectors)

✅ External exposure

---

## \*\*2. Key Concepts\*\*

### \*\*1. Service Discovery & Load Balancing\*\*

- \*\*Labels & Selectors:\*\*

- Services track Pods using \*\*labels\*\* (e.g., `app: payment`).

- Example:

```yaml

# Pod Label

metadata:

labels:

app: payment

# Service Selector

spec:

selector:

app: payment

```

- \*\*How It Works:\*\*

- Service forwards traffic to \*\*all Pods\*\* with matching labels.

- If a Pod dies, ReplicaSet creates a new one with the \*\*same label\*\*.

### \*\*2. Service Types\*\*

| \*\*Type\*\* | \*\*Use Case\*\* | \*\*Accessibility\*\* |

|----------------|---------------------------------------|---------------------------------|

| \*\*ClusterIP\*\* | Internal communication (default) | Only within Kubernetes cluster |

| \*\*NodePort\*\* | Access from within organization/VPC | Via worker node IPs |

| \*\*LoadBalancer\*\*| Public access (cloud providers) | External users (e.g., internet) |

![Service Types](https://d33wubrfki0l68.cloudfront.net/701651737efd9e6a0b9d0c0106eb441e7b82b6d7/9c9b6/docs/tutorials/kubernetes-basics/public/images/module\_04\_services.svg)

---

## \*\*3. Hands-On: Creating a Service\*\*

### \*\*Step 1: Create a Deployment\*\* (`deployment.yaml`)

```yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

```

### \*\*Step 2: Create a Service\*\* (`service.yaml`)

#### \*\*Option A: ClusterIP (Internal)\*\*

```yaml

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

type: ClusterIP

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

```

#### \*\*Option B: NodePort (Internal + VPC)\*\*

```yaml

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

type: NodePort

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

nodePort: 30007 # Optional (30000-32767)

```

#### \*\*Option C: LoadBalancer (Public)\*\*

```yaml

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

type: LoadBalancer

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

```

### \*\*Step 3: Apply Configs\*\*

```sh

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

```

### \*\*Step 4: Verify\*\*

```sh

kubectl get svc # List services

kubectl describe svc nginx-service # Check details

```

---

## \*\*4. Key Commands\*\*

| \*\*Task\*\* | \*\*Command\*\* |

|---------------------------|--------------------------------------|

| List Services | `kubectl get svc` |

| Debug Service | `kubectl describe svc <name>` |

| Access ClusterIP Service | `curl <cluster-ip>:80` (inside cluster) |

| Access NodePort Service | `curl <node-ip>:30007` (from VPC) |

| Access LoadBalancer | `curl <external-ip>:80` (public) |

---

## \*\*5. Interview Questions\*\*

### \*\*Q1: Why use Services over direct Pod IPs?\*\*

- Pod IPs are \*\*ephemeral\*\* (change on restart).

- Services provide \*\*stable endpoints\*\* + load balancing.

### \*\*Q2: How does Service Discovery work?\*\*

- Services use \*\*labels/selectors\*\* to track Pods (not IPs).

### \*\*Q3: Difference between NodePort and LoadBalancer?\*\*

- \*\*NodePort:\*\* Exposes on worker node IPs (internal/VPC).

- \*\*LoadBalancer:\*\* Creates cloud LB (public access).

---

## \*\*6. Summary\*\*

- \*\*Services solve:\*\*

- Load balancing (traffic distribution).

- Service discovery (label-based tracking).

- External exposure (public/private access).

- \*\*Always use Services\*\* (not direct Pod IPs) in production.

🚀 \*\*Assignment:\*\*

1. Deploy an app with a \*\*ClusterIP\*\* Service.

2. Modify to \*\*NodePort\*\* and test access.

3. (Optional) Try \*\*LoadBalancer\*\* on cloud (AWS/GCP).

📌 \*\*Pro Tip:\*\* Use `kubectl expose` for quick Service creation:

```sh

kubectl expose deploy nginx-deployment --port=80 --type=NodePort

```

📢 \*\*Feedback?\*\* Comment below! 👍 \*\*Like & Share\*\* if this helped!

---

\*\*End of Notes\*\* 🎉